NAS2-12961
IN-61-CR
026 183

NAS Kernels Survey Report

David Browning[1]

Report RND-92-003 February 1992

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

ARC 275 (Rev Feb 81)

NAS Kernels Survey Report

David Browning[1]

Report RND-92-003 February 1992

NAS Systems Development Branch
NAS Systems Division
NASA Ames Research Center
Mail Stop 258-6
Moffett Field, CA 94035-1000

[1] Computer Sciences Corporation, NASA Contract NAS 2-12961, Moffett Field, CA 94035

NAS Kernels Survey Report

David Browning
Computer Sciences Corporation
NASA Ames Research Center
Moffett Field, CA 94035, USA

Abstract

The NAS Kernels Benchmark Program (NASKER) has been used to measure performance on the Cray C90, Cray Y-MP, Cray-2, Alliant FX/2800 and Connection Machine 2 computer systems. The ground rules for this test are reiterated, with some changes from the original proposal. On each system, tests are performed using varying numbers of processors. Performance results are presented at various tuning levels, and code modifications are described. Differences between this benchmark program and the NAS Parallel Benchmarks are discussed.

Introduction

The NAS Kernels Benchmark Program (NASKER) [3] is a benchmark program developed at NAS for the evaluation of supercomputer performance. The benchmark is comprised of seven computational kernels, chosen to be representative of the Computational Fluid Dynamics workload at NAS. This report surveys the performance of the NAS Kernels on a variety of computer systems.

The NAS Kernels

NASKER is described at length in [3], which also includes a listing of the Fortran source code. Briefly, the seven kernels are as follows:

1. MXM — Outer product matrix multiplication.
2. CFFT2D — 2-dimensional complex Fast Fourier Transforms.
3. CHOLSKY — Simultaneous banded Cholesky factorization and solution.
4. BTRIX — Simultaneous block tridiagonal matrix solutions.
5. GMTRY — Vortex method solution using Gaussian elimination.
6. EMIT — Creation of new vortices according to certain boundary conditions.
7. VPENTA — Simultaneous pentadiagonal matrix solutions.

After execution, NASKER prints the computational error, number of floating point operations, execution time and performance for each kernel and for the whole program. Details of these calculations are provided in [3].

Although each kernel emphasizes a supercomputer's vector performance, taken together they were designed to not favor a particular vector architecture. They include a variety of calculations and memory operations, and contain loops with a range of vector lengths.

NASKER also tests the capabilities of vectorizing compilers. Over 99% of the floating point calculations are found in potentially vectorizable loops, but some are challenging to vectorize. To show the hardware capability as well as the the amount of programmer intervention required, results are published with varying levels of optimization. This demonstrates the maximum performance possible on the kernels, while showing how much programming effort is required to achieve these results.

Ground Rules

There are four levels of optimization permitted when presenting results, levels 0, 20, 50 and unlimited. Each level corresponds to the maximum number of lines of code that may be changed, added or deleted. The addition of a compiler directive is considered a change. There is one exception, the timing routine CPTIME, which may be modified without penalty as described in the source listing.

The benchmark program may be split into separate source files. Each source file may then be compiled separately, with arbitrary compiler options allowed on each source file. This customization does not count toward the limits prescribed by each tuning level. The issue of separate source files and different compiler options was not explicitly addressed in [3], though it was probably assumed that the benchmark program would be compiled as a single source file. The common use of makefiles in scientific programming today makes it reasonable to allow this "command line" tuning.

Originally, [3] did not allow calls to external subprograms unless they were included in the program file and satisfied other requirements, such as ANSI FORTRAN-77 conformance [1]. Today, however, optimizing compilers are capable of replacing certain code constructs with calls to external, non-standard, highly optimized library calls, without user intervention. Furthermore, NASKER is being run on parallel computers, where non-standard Fortran constructs and library calls, such as those required for interprocessor communication, are essential. For these reasons, certain extensions to FORTRAN-77 are now permissible. In particular, constructs contained in the Fortran 90 [6] and Parallel Computing Forum [8] drafts are acceptable, as are other constructs which facilitate execution on multiple processors. For example, extensions and library routines which indicate parallel regions of code, specify allocation or organization of data among processors, communicate data between processors, or synchronize the action of processors are allowed. In addition, computational routines which perform dense matrix-matrix or matrix-vector multiplication, or onedimensional or two-dimensional fast Fourier transforms are allowed. Such routines must be callable with general array dimensions. Any extension to FORTRAN-77 must be part of the vendor's usual library and available in every installation.

The rules have also changed with regard to measuring multiprocessor performance with the NAS Kernels. In [3], it was suggested that NASKER be run simultaneously on each processor of a parallel computer. This would measure the amount of interprocessor resource contention, but only on a shared memory system. The approach in this report is to distribute a single instance of the benchmark program over all or part of the parallel computer. As with vectorization in the single processor test, this measures the ability of the compiling system to automatically detect and exploit parallelism. Unlike the approach suggested in [3], this demonstrates the impact of communication overhead on distributed memory machines, as well as memory and bus contention on shared memory machines. There are plans to port the NAS Kernels to distributed memory, message passing architectures.

Results

In Figure 1, the performance results are shown for the machines most recently tested, at various tuning levels and numbers of processors. The performance (in millions of floating point operations per second, or MFLOPS) is shown for each kernel, using the results printed by the benchmark program during execution. See [3] for a discussion on how these rates are determined. Except for the C90, all tests with more than one processor were done in a dedicated mode with no other users sharing the processors at the time, and were completed during June and July 1991. All vendors represented in this survey were given the opportunity to provide source code (and compiler options) at various tuning levels. A previous survey presented results for the Cray X-MP, CDC 205 and Amdahl 1200 computer systems [2].

Cray Y-MP/8128

These tests were performed by the author on the machine named Reynolds at NAS. It had the following configuration at the time of the tests:

Processors: 8

Main Memory: 128 MWords (1 GByte)

Compiler: CF77 5.0, Field Test Version

Operating System: UNICOS 6.0

Source code, including makefile, for both 0-line and 20-line tuning levels were supplied by Cray Research, Inc. [7]. The code was compiled and run by the author. Although any number of processors may be allocated at run time, only powers of two were compatible with the tuning changes made to the level-20 code. For comparison purposes, only powers of two were allocated for the level-0 code as well.

Each version of NASKER was compiled with all phases of the Cray compiling system invoked. This includes a dependence analyser ("fpp") which inserts directives for Autotasking and vectorization, and a translator ("fmp") which converts the Autotasking directives into Fortran code. The final phases are compiling ("cft77") and loading ("segldr"). The program was recompiled for each number of processors, since the compiling system makes use of this information through command line options. In the level-20 tuning, some changes to the source code included the number of processors to be used, so different versions of the level-20 code were used for different numbers of processors.

On a single processor, the level-20 version showed significant improvements (performance increase of 10% or more) in the MXM, CFFT2D and VPENTA kernels. Exchanging two lines in MXM leads to an "inner product" algorithm, which is not necessarily more efficient, but is recognized by the compiler as a matrix-multiplication and automatically replaced by a library call. One line of source in each of CFFT2D and VPENTA were changed to alter the dimensions of the arrays, eliminating power-of-two strides and reducing memory bank conflicts.

The kernels CFFT2D, CHOLSKY, BTRIX, EMIT and VPENTA had microtasking directives ("CMIC$ DO ALL . . .") added in the level-20 version to improve multiprocessor performance. Code was added to CFFT2D, CHOLSKY, and VPENTA to distribute work among processors. Figures 2 and 3 show the speedups obtained for level-0 and level-20 versions of the program. (If R_n is the performance rate for n processors reported in Figure 1, then the speedup S_n graphed in Figures 2–9 is given by $S_n = R_n/R_1$.)

Cray-2/4-256

These tests were performed by the author on the machine named Navier at NAS. It had the following configuration at the time of the tests:

Processors: 4
Main Memory: 256 MWords (2 GBytes)
Compiler: CF77 5.0, Field Test Version
Operating System: UNICOS 6.0

Cray Computer Corporation did not supply their own version of the benchmark program [11], so the same versions used on Reynolds were used on Navier. Both Reynolds and Navier used the same compiling system.

As on the Y-MP, optimizations in the level-20 version improved single-processor performance in the MXM, CFFT2D and VPENTA kernels. The differences were much more dramatic on the Cray-2, however. In the case of the VPENTA kernel, the performance increased by over 16 times. This shows the extreme sensitivity of the Cray-2 to power-of-two strides.

Figures 4 and 5 show the speedups attained for each kernel when run on 2 and 4 processors. Speedups are not nearly as good as those attained by the Y-MP, even for the level-20 versions of the program. In fact, the MXM kernel showed no speedup when run with 2 processors, and the performance of the VPENTA kernel actually decreased from 2 processors to 4 processors. The reason for this behavior is not known.

Cray Y-MP C90/16256

These tests were performed by CRI personnel, not the author. The system configuration reported by CRI is as follows:

Processors: 16
Main Memory: 256 MWords (2 GBytes)
Compiler: CF77 5.0
Operating System: UNICOS 7.0

(In this report, "Y-MP" refers to the Cray Y-MP/8128, while "C90" refers to the Cray Y-MP C90/16256.) CRI reports that the C90 tests were performed with the same code they supplied for the Y-MP tests, but were not run in a dedicated mode. The C90 tests were performed in October 1991. [7]

The performance increase of a single C90 processor over a single Y-MP processor varies considerably for the different kernels. The smallest improvement occurs with the level-0 version of the VPENTA kernel, the C90 performing less than 13% faster than the Y-MP. However, the C90 processor performs 2.8 times faster than the Y-MP processor on the level-20 version of the same kernel. On both CFFT2D and VPENTA, the effect of removing power-of-two strides is more pronounced on the C90 than the Y-MP, but not to the degree of the Cray-2.

With 8 processors running the level-20 version of MXM, the C90 achieved a speedup of 7.6, compared to the speedup of 7.1 attained by the Y-MP. Note that these speedups are achieved using a highly optimized library call which is automatically invoked by the CF77 compiler. The speedups attained by the C90 are shown in Figures 6 and 7.

CM-2

These tests were performed by the author on the Connection Machine 2 at NAS. It had the following configuration at the time of the tests:

Processors: 32,768 (32K)
Main Memory: 2 GBytes
Compiler: CM Fortran 1.0

The 32K 1-bit processors are supplemented by 1024 64-bit floating point units. Each processor has 128K bytes of memory, for a total of 2 GB of memory in the CM-2. The front-end processors are a Sun 4/490 ("Simeon") and a VAX 6230

("Gottfried"), each with two sequencer interface boards. Simeon was used for these tests.

The version of NASKER used on the CM-2 was developed at NAS by Russell Carter [5]. Carter reported on the effort required to port NASKER to CM Fortran, and measured the performance of the ported code. The code required significant modification to force computations to occur on the CM-2 instead of the front-end, and is therefore in the level-unlimited tuning category. The number of repetitions of each kernel was reduced to the minimum to keep execution time reasonable, and performance output was printed with more decimal digits. (Performance rates were not affected by these changes.) The MXM kernel was modified to call the matrix multiplication library routine MATMUL. Thinking Machines Corporation was invited to further tune the code in preparation of this report, but did not. [10]

At the time [5] was prepared, the CM-2 had 32-bit floating point units. With the new hardware and compiler, which allows 64-bit arithmetic and "slicewise" data storage, every kernel shows improved performance over that reported in [5]. Unfortunately, the errors reported by NASKER have not improved. According to [3], the total of the relative errors from the seven kernels must be less than 5×10^{-10}. The errors reported for three of the kernels, CFFT2D, GMTRY and EMIT, were greater than 10^{-8}. This suggests that some arithmetic, probably with complex data, is not being performed with 64-bit precision. This was also the case in [5].

Figure 8 shows the speedups attained with 16K and 32K processors over 8K processors.

Alliant FX/2800

These tests were performed by Alliant personnel, not the author. The machine configuration reported by Alliant is as follows:

Model 200 Processor Modules (uses Intel i860XR 40 MHz processors)
4 MB Global Cache
FX/Fortran version 1.2.0

Alliant personnel supplied the author with source code, makefiles, and program output, along with explanations of the code modifications [9].

Although all floating point arithmetic was done with 64-bit precision, default integer word length is 32-bits. In the GMTRY and EMIT kernels, the declarations of integer arrays in common blocks were modified so that subsequent real arrays were aligned on 64-bit word boundaries. In the kernels CFFT2D and VPENTA, array dimensions were modified to improve memory access of the interleaved cache banks. In CHOLSKY, certain array indices were transposed to obtain stride-one addressing in inner loops and improve local memory access patterns. Only the GMTRY and VPENTA kernels exhibited significant performance improvement under these modifications when run on a single processor. The effect of these changes upon multiprocessor performance cannot be assessed, since 0-level results on 14 processors were not reported.

The speedups attained with 14 processors are shown in Figure 9, using the level-20 version of the program. Surprisingly, the VPENTA kernel shows a "perfect" speedup of 14. One possible explanation is that decomposition and distribution of the work in VPENTA led to greater data locality in each processor and better use of the cache.

Conclusion

As long as the NAS computational workload is represented to some extent by these kernels, NAS will continue to use this benchmark program in evaluating supercomputer performance. Proponents of highly parallel computer systems argue that these kernels do not demonstrate the full performance capabilities of parallel computers. This is indeed true. The kernels lack sufficient parallelism for machines such as the CM-2, the algorithms may be poorly matched to a given parallel architecture, and the use of parallel language constructs require higher tuning levels. However, these arguments do not imply that the NASKER program should not be run on such machines. Instead, potential users should be aware that they will not achieve good performance on programs with similar computational kernels.

The NAS Parallel Benchmarks [4] avoid these difficulties on highly parallel systems. The problems are larger, and are specified only in definition of the problem, not in implementation. This allows vendors greater freedom to demonstrate high performance on problems relevant to NAS, the cost being greater development effort. However, performance on the NAS Parallel Benchmarks does not necessarily reflect performance on existing CFD programs. The NAS Kernels Benchmark Program is designed to measure the latter.

COMPUTER	NO. CPUS	TUN- ING	MXM	CFFT	CHOL	BTRX	GMTRY	EMIT	VPNTA	TOTAL
Cray Y-MP/8128	1	0	276.04	76.53	90.37	140.86	251.32	181.63	51.39	108.88
	1	20	304.99	192.12	90.45	150.92	251.15	181.10	196.79	180.84
	2	0	547.72	136.43	145.21	200.34	484.27	355.06	72.28	177.50
	2	20	600.16	351.67	163.23	290.92	484.28	355.58	382.17	341.68
	4	0	1060.80	206.82	208.21	253.44	921.33	681.07	72.97	234.40
	4	20	1196.14	577.42	273.10	527.92	922.17	679.50	649.43	601.76
	8	0	1896.36	261.63	209.74	271.02	1634.36	1217.02	72.62	262.90
	8	20	2176.50	778.31	383.29	979.86	1622.14	1236.62	950.58	931.10
Cray-2/4-256	1	0	187.50	11.24	27.15	44.40	176.52	147.52	9.28	23.43
	1	20	416.53	65.30	27.15	45.02	176.75	148.47	155.81	76.50
	2	0	174.82	18.38	37.17	47.03	288.55	269.10	9.78	30.84
	2	20	412.98	84.21	50.25	81.63	288.59	271.31	288.61	122.12
	4	0	394.86	19.90	44.65	50.41	421.34	365.53	9.76	33.33
	4	20	728.56	161.76	55.78	117.03	347.23	369.40	240.20	170.89
Cray C-90/16256	1	0	787.21	97.09	244.17	273.44	672.38	407.50	57.86	165.57
	1	20	914.03	473.25	297.96	272.32	669.44	405.64	553.88	452.65
	2	0	1542.29	171.27	374.88	341.66	1256.31	788.33	78.49	256.06
	2	20	1714.78	592.94	276.02	497.40	1224.92	708.91	834.69	649.03
	4	0	3008.06	257.72	517.81	422.50	2347.05	1516.70	78.80	319.54
	4	20	3531.74	1361.20	809.46	1009.40	2345.64	1521.34	1144.30	1402.90
	8	0	4503.12	321.97	506.62	436.15	3776.99	2693.62	77.68	344.93
	8	20	6921.63	1723.45	1011.19	1699.24	3894.25	2732.61	1304.99	1979.77
CM-2	8K	unlim	3.464	0.066	0.033	0.006	2.144	0.530	0.064	0.49
	16K	unlim	6.100	0.107	0.055	0.010	2.966	0.828	0.116	0.77
	32K	unlim	11.210	0.196	0.096	0.015	5.475	1.552	0.121	1.29
Alliant FX/2800	1	0	21.87	7.09	3.93	4.56	2.13	10.28	3.18	5.10
	1	20	21.92	7.12	3.95	4.58	4.97	10.44	3.61	6.13
	14	20	247.78	34.35	15.05	40.89	38.39	77.99	50.76	41.25

Figure 1. NAS Kernel Benchmark Program (MFLOPS)

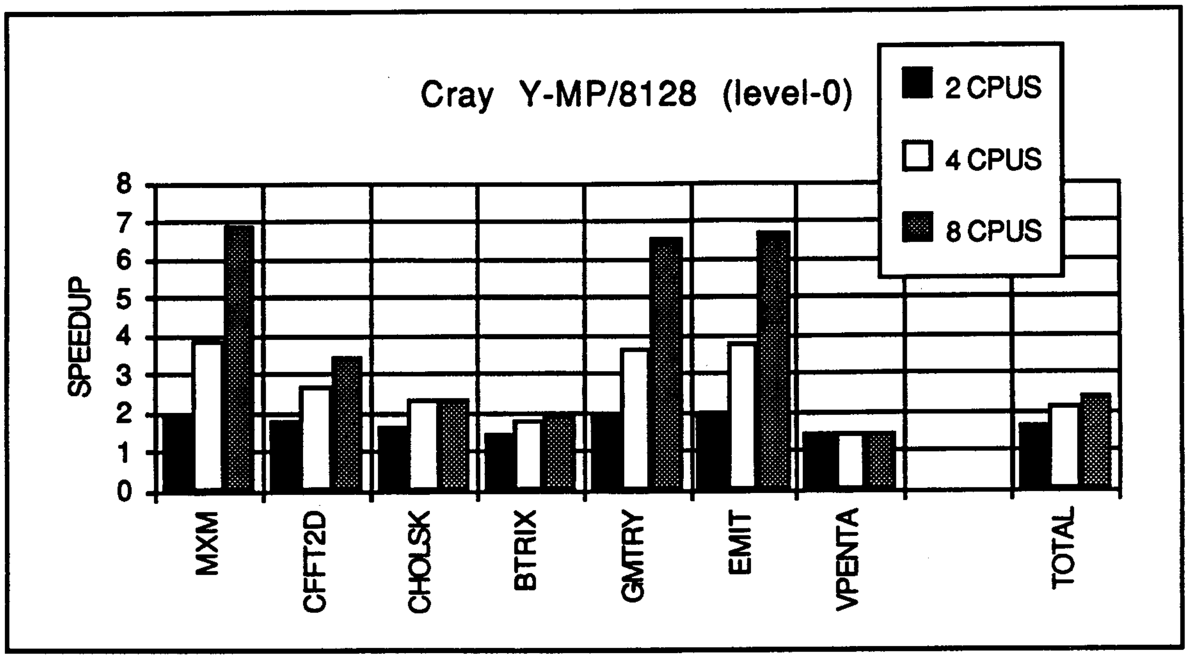

Figure 2. Speedup of level-0 NAS Kernels on Y-MP

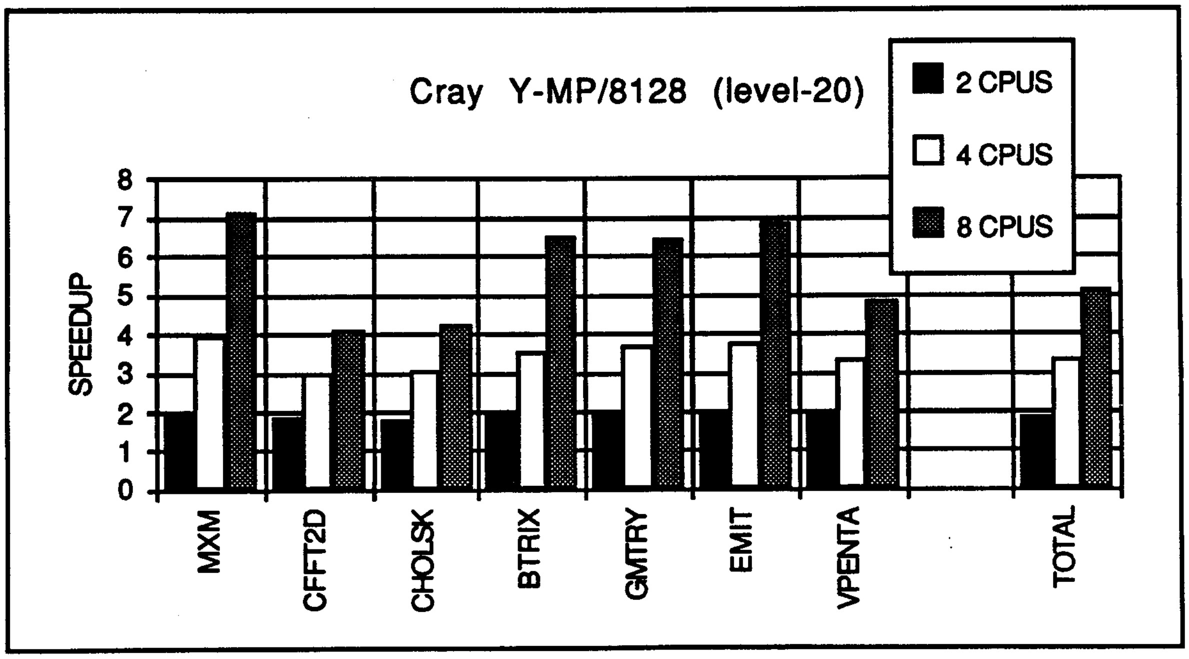

Figure 3. Speedup of level-20 NAS Kernels on Y-MP

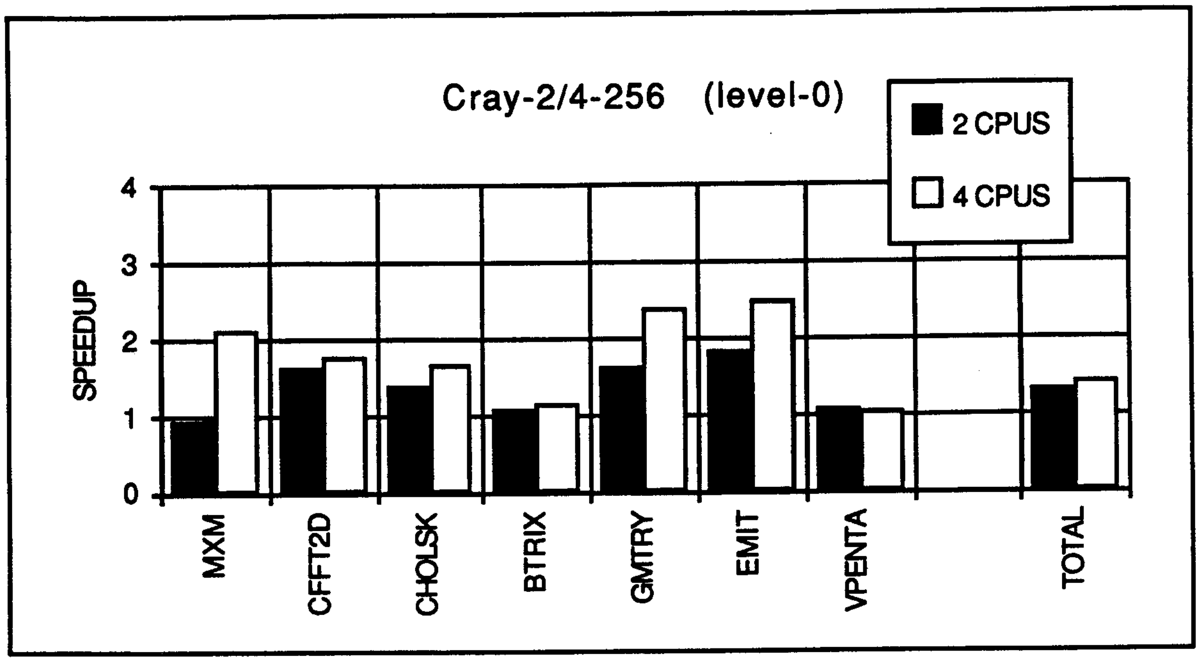

Figure 4. Speedup of level-0 NAS Kernels on Cray-2

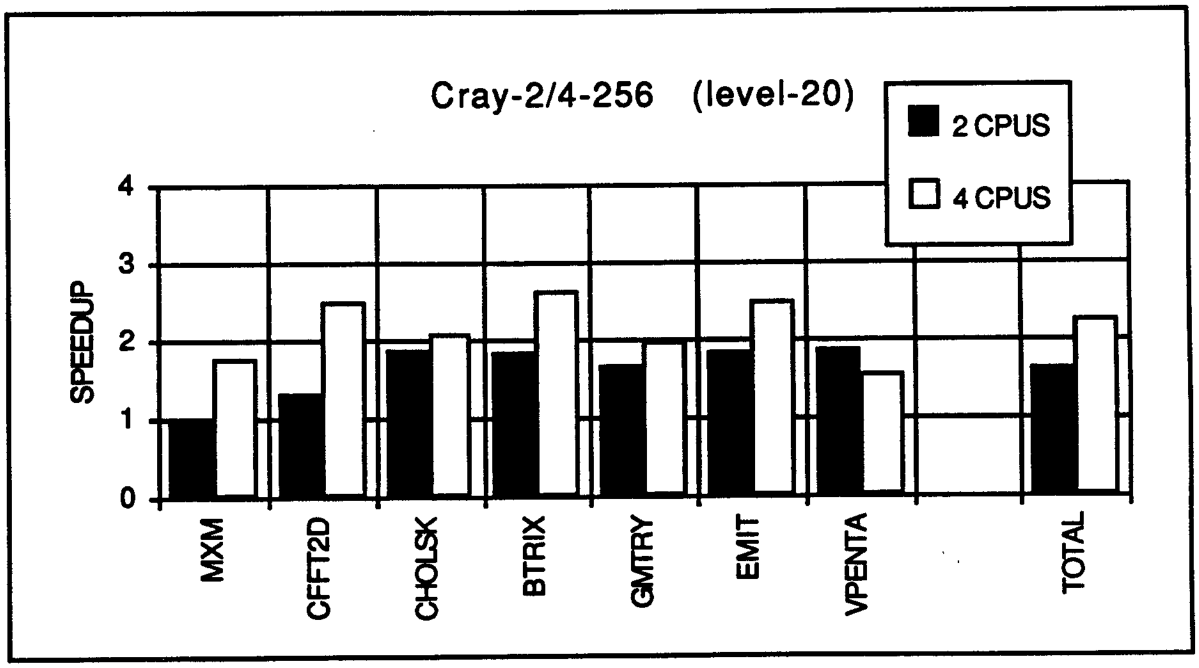

Figure 5. Speedup of level-20 NAS Kernels on Cray-2

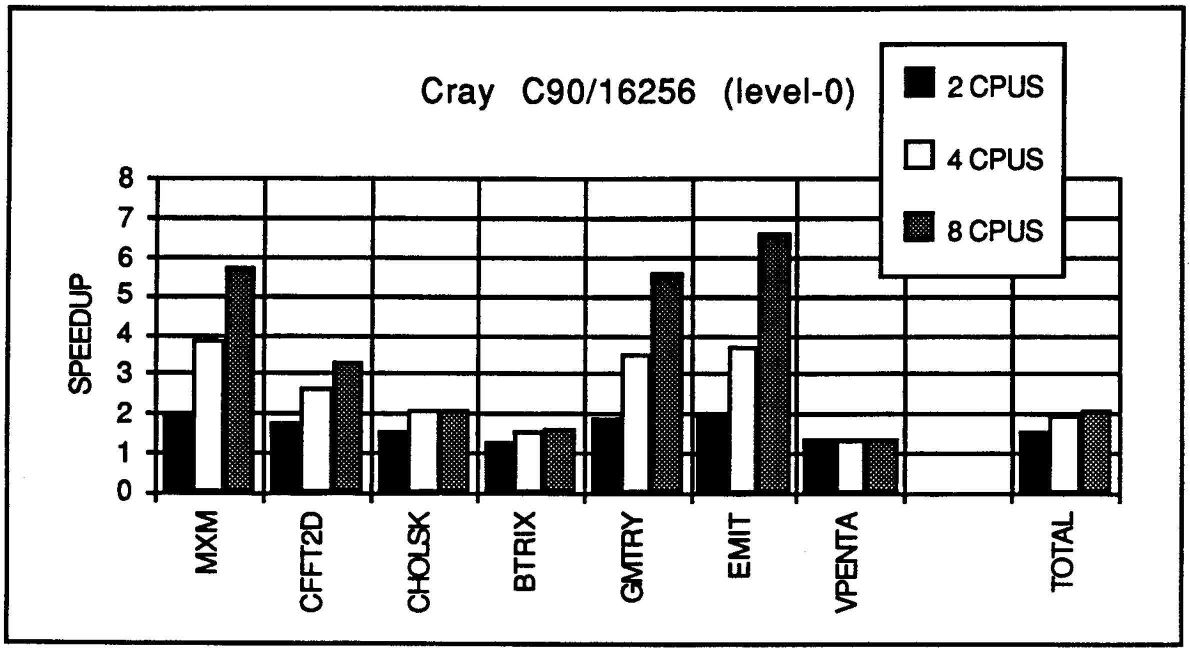

Figure 6. Speedup of level-0 NAS Kernels on C90

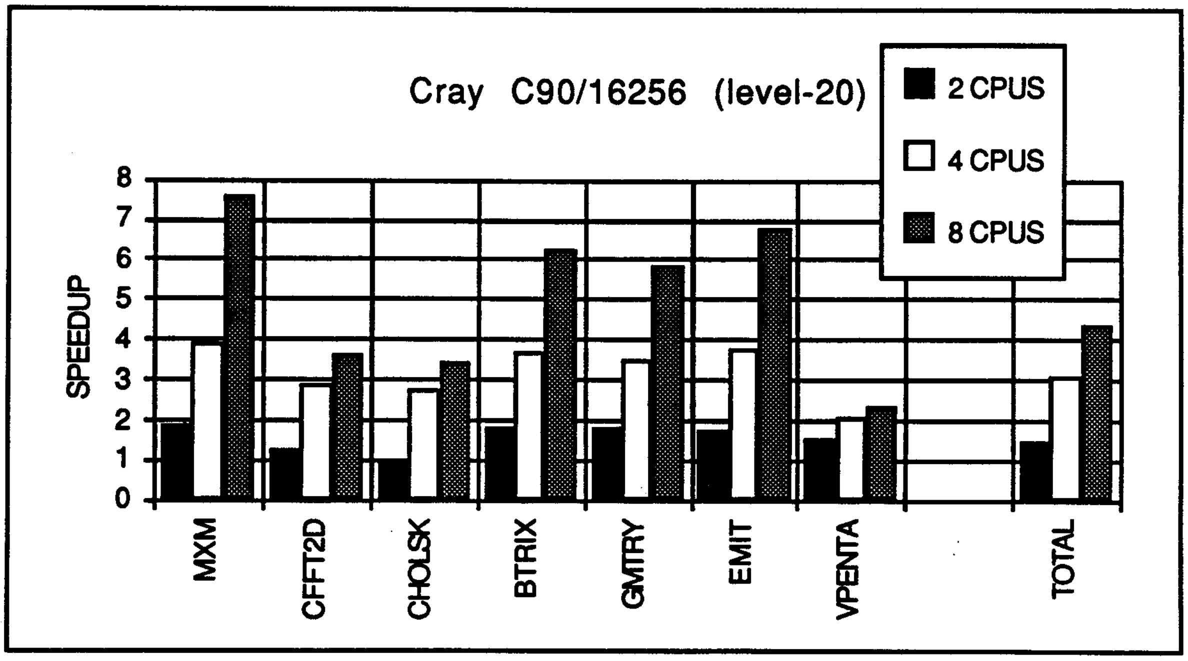

Figure 7. Speedup of level-20 NAS Kernels on C90

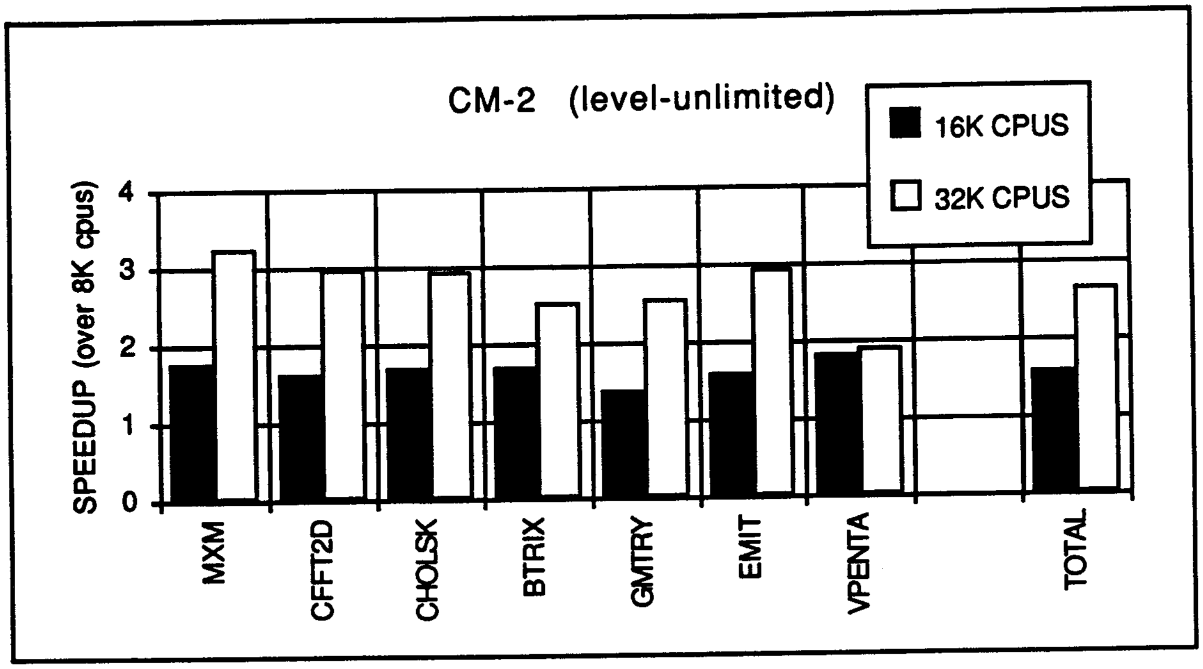

Figure 8. Speedup of level-unlimited NAS Kernels on CM-2

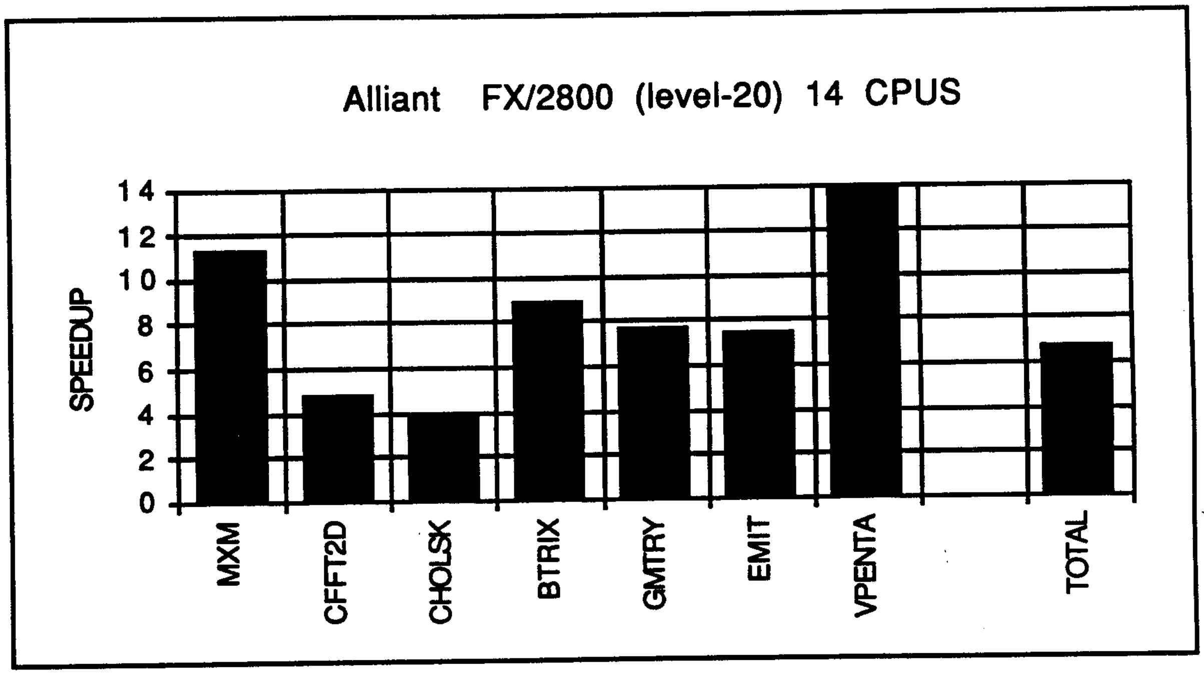

Figure 9. Speedup of level-20 NAS Kernels on Alliant FX/2800

References

[1] *American National Standard Programming Language FORTRAN, ANSI X3.9-1978,* American National Standards Institute, Inc., 1978.

[2] D. H. Bailey, "NAS Kernel Benchmark Results," *First International Conference on Supercomputing Systems,* IEEE Computer Society, 1985, pp 341-345.

[3] D. H. Bailey and J. Barton, "The NAS Kernel Benchmark Program," NASA Technical Memorandum 86711, August 1985.

[4] D. H. Bailey, J. Barton, T. Lasinski, H. Simon, eds., "The NAS Parallel Benchmarks," NAS Report RNR-91-002, January 1991.

[5] R. Carter, "NAS Kernels on the Connection Machine," NAS Report RND-90-005, August 1990, revised February 1991.

[6] "Fortran 90," Draft S8 Version 112, X3.9-198x Draft Proposed American National Standard Fortran, American National Standards Institute, Inc., 1989.

[7] C. Grassl, Cray Research, Inc., personal communication, June 1991 and October 1991.

[8] "PCF Parallel Fortran Extensions," Version 5.0, Parallel Computing Forum, c/o Kuck & Associates, July 1991.

[9] J. Perry, Alliant Computer Systems Corporation, personal communication, July 1991.

[10] J. Richardson, Thinking Machines Corporation, personal communication, June 1991.

[11] H. Watts, Cray Computer Corporation, personal communication, June 1991.